# Docker基础



## Docker概述





### 1.为什么会出现Docker?

传统项目开发中遇到的问题：

1.环境部署配置麻烦;一个项目从开发到上线一般会经历多套环境部署（开发环境、测试环境、生产环境），其中环境的部署、应用的配置都是非常麻烦的，在服务器上配置应用的运行环境,需要安装各种软件(Jdk,tomcat,nginx,mysql等),软件的安装和配置不仅麻烦且容易出错；由于互联网高并发环境和大流量的访问基本都是集群环境(多台机器)每一台机器都要部署环境，容易出错且费时费力。

2．应用环境的部署和配置不能跨平台;在window上部署了环境到Linux上又 重新部署一遍;

3.应用的移植性差;即使换另一台同样操作系统的服务器，要移植应用也是非常麻烦的；

4.应用隔离开销大; 不同的应用程序依赖的环境不一样，如果我们把依赖的环境安装在一台服务器上,配置麻烦不说而且可能会出现冲突（端口冲突），如果想把两个应用程序隔离起来，需要在服务器上创建不同的虚拟机，不同的虚拟机放不同的应用，但是虚拟机的开销比较高；

5开发和运维之间协作需要关心的问题多;传统开发中,软件开发-测试完成后，为了让程序可以顺利在生产环境下部署，开发需要将代码（jar，war）交给运维人员,并准备完整的部署文档，清楚的告诉运维人员，用的全部配置文件及所有软件环境,让运维得以顺利部署应用，在软件交付的过程中由于环境和配置的不同经常出现部署失败问题；

6.运维考验大;在项目的开发和维护阶段，由于各种版本的迭代，随着代码和配置越来越复杂,稍有不慎就可能导致对应的服务不可以使用,解决不同版本环境的兼容，对运维的考验较大；

Docker提供了一套标准化的解决方案，解决了运行环境和配置问题,方便做持续集成及发布,为应用的开发 、运行和部署提供了“ 一站式”的解决方案。

传统的项目环境部署麻烦、开发和运维之间协作关心的事情多;Docker可以将我们项目带上环境安装打包部署，支持将项目打包编译为镜像，对于运维来说只需要发布镜像，运行镜像就相当于运行项目，简化对Linux的操作成本,降低了运维的运维难度；

运维考验大,由项目迭代、环境和配置不同可能导致服务不可以用;Docker的镜像提供了除内核外完整的运行时环境，确保了应用环境一致性。Docker使用了分层存储以及镜像技术，使得应用的维护更新更加的简单;

环境部署和配置麻烦、不能跨平台且移植性差，在集群环境下，每一台机器都需要安装对应的软件(数据库、tomcat、等);Docker可以将应用程序需要的系统运行环境，由下而上打包为镜像,其他机器只需要下载镜就可以使用，运行镜像就相当于运行我们的软件运行环境,实现应用程序跨平台运作，让运维更加简单;

### 2.什么是Docker？

Docker是使用 Go语言实现的开源容器项目，是一种解决了运行环境和配置问题的软件容器,方便做持续集成并有助于整体发布的容器虚拟化技术。诞生于2013年,最初发起者是dotCloud公司；

Docker的目标是通过对应用组件的封装、分发、部署、运行等生命周期的管理，使得我们的应用及其运行环境能够做到”一次封装，到处运行”;Linux容器技术的出现解决了这个问题,Docker是在Linux容器基础上发展过来的。将应用运行在Docker容器上，由于Docker容器在任何操作系统上都是一致的,这样就实现了跨平台、跨服务器。只需要一次配置好环境，换到其他机器上就可以一键部署好,大大简化了操作;

Docker 项目已加 Linux 基金会，并遵循Apache协议,全部开源代码在 <https://github.com/docker> 项目仓库进行维护;

目前主流的操作系统包括 Linux 各大发行版、 macOS , Windows 等都 支持 Docker;



### Docker与虚拟机的区别





在容器技术出现之前，我们都是使用虚拟机技术;

**什么是虚拟机？**

虚拟机是一个软件,通过软件(Vmware、Virtual Box等软件)可以虚拟出一台或者多台具有完整硬件功能的、运行在一个完全隔离的环境中的完整操作系统(电脑)，在实体计算机中能够完成的工作在虚拟机中都能够实现,且在虚拟机中的操作不会对真正的系统产生任何影响;

**虚拟机是带环境安装的一种解决方案，可以在一种操作系统中运行另一种操作系统;**如在windows系统上运行Linux系统,应用程序对此毫无感知，因为虚拟机看上去和真实系统一模一样，而对底层系统来说，虚拟机只是一个普通文件，不需要就删掉,对其他部分毫无影响,这类虚拟机完美的运行了另外一套操作系统，能够使应用程序、操作系统和硬件三者之间的逻辑不变;

虚拟机技术是虚拟化技术的一种;

**传统的虚拟机技术的优点：**

1.通过虚拟机虚拟出来的操作系统（电脑）,具有真实系统（电脑）完全一样的功能;

2.进入虚拟机系统后，所有的操作都是在虚拟机系统里进行（安装软件，数据保存等），拥有自己独立的桌面，不会对真正的操作系统造成任何影响且能够在现有系统与虚拟镜像之间灵活切换；

3.通过虚拟机虚拟出来的每一个操作系统（每台电脑）之间是完全相互隔离的;

**传统的虚拟机技术的缺点:**

传统虚拟机技术是在硬件层面实现虚拟化的,需要模拟底层的硬件指令,启动速度相对较慢；

虚拟机的体积较大,1个虚拟机可能就十多个G了;

虚拟机占用系统资源多,会占用系统很多的内存;

虚拟机的使用步骤冗余需要开机且开机过程复杂;

虚拟机的启动慢,一般需要几分钟；

**Docker容器虚拟化技术：**

由于虚拟机的缺点,Linux发展出了Linux容器，Linux容器不是模拟一个完整的操作系统,而是对进程相互隔离。有了容器，可以将软件运行时所需要的所有资源打包到一个隔离的容器中。容器与虚拟机不同，不需要绑定一整套操作系统，只需要软件工作所需要的库资源和设置。系统因此变得更加高效轻量级并保证保证部署在任何环境中的软件都能始终如一的运行。Docker容器技术也是在Linux容器的基础上发展过来的

Docker容器技术也是一种虚拟化技术,容器化技术不是模拟的一个完整的操作系统;

Docker容器是在操作系统层面实现虚拟化的,模拟的只是一个最核心的环境.Docker与宿主机共享硬件资源及操作系统,是内核级别的虚拟化,容器与容器之间共享内核因此更加轻便（是MB级别的）;

**相同点：**

1.虚拟机也是属于虚拟化技术,Docker容器技术也是一种虚拟化技术；

**区别：**



传统虚拟机方式是在硬件层面实现虚拟化，需要有额外的虚拟机管理应用和虚拟机操作系统 层。虚拟出一套硬件后，运行一个完整的操作系统，然后在这个系统上安装和运行软件。体积比较笨重，占用系统资源较多（一个虚拟机对硬盘的占用是GB级别的）,启动速度十分缓慢(是分钟级别的);

Docker 容器是在操作系统层面上实现虚拟化，直接复用本地主机的操作系统，因此更加轻量级;容器内的应用直接运行在宿主机的内核中，容器是没有自己的内核的,而且没有对硬件虚拟，因此容器比传统的虚拟机更为轻便;每个容器之间是相互隔离的，每个容器内都有一个属于自己的文件系统，容器之间进程不会互相影响，能区分计算资源;

Docker容器无需虚拟硬件，模拟的只是一个最核心的环境，体积小巧,占用资源少(一个容器镜像对硬盘的占用是MB级别的)，启动速度非常快（可以达到秒级别）；

相关资料：

<https://www.jianshu.com/p/d3006b8a22ee>

<https://www.cnblogs.com/maji233/p/11032392.html>

<https://www.zhihu.com/question/48174633>

### 4.Docker能做什么？

1.开发/运维(Devops),开发及运维，实现一次构建、随处运行;实现更快速的应用交付和部署项、更便捷的升级和扩缩容、更简单的系统运行、更高效的计算资源利用；

2. 容器技术使得大规模动态调度成为可能；(12306余票查询峰值问题，京东618促销峰值浏览问题，新闻热点峰值问题)

3.应用程序的打包和发布

4.应用程序隔离

5.自动化测试、持久集成、发布

6.部署微服务

7.快速搭建测试环境

8.提供PaaS平台级产品

### 5. 为什么使用Docker？

;

Docker作为一种新兴的虚拟化技术相对传统的虚拟化方式相比有着众多的优势:

1. 简单，易用，活跃的社区;

Docker使用简单,容易上手，有活跃的社区;

**2.更高效的利用系统资源;**

可以在一台物理机上运行很多的容器实例，由于容器不需要进行硬件虚拟以及运行操作系统等额外开销,Docker对系统资源的利用率更高。

无论是应用执行速度、内存损耗或者文件存储速度，都要比传统虚拟机技术更高效, 尤其在内存和IO的性能开销更小；

**3.更快的启动时间；**

轻级，秒级的快速启动速度,传统的虚拟机技术启动应用服务往往需要数分钟，而Docker容器应用，由于直接运行在宿主内核，无需启动完整的操作系统,因此可以做到秒级、甚至毫秒级的启动时间。大大节约了开发、测试、部署时间;

**4.一致的运行环境；**

开发中一个常见的问题是环境一致性问题。由于开发环境、测试环境、生产环境不一致，导致有些bug并未在开发过程中被发现。而Docker的镜像提供了除内核外完整的运行时环境，确保了应用环境一致性。

**5.更快速交付和部署;**

易于构建应用，有良好的Rest API，很适合自动化测试和持续集成；使用Docker可以通过定制应用镜像来实现持续集成、持续交付、部署。

标准统一的打包、部署、运行方案；开发人员可以通过Dockerfile来进行镜像构建，并结合集成系统进行集成测试，而运维人员可以直接在生产环境中快速部署镜像，甚至结合持续部署系统进行自动部署。

镜像支持增量分发、易于部署；

**6.更轻松的迁移；**

由于Docker确保了执行环境的一致性，使得应用的迁移更加容易。Docker可以在很多主流平台上运行;

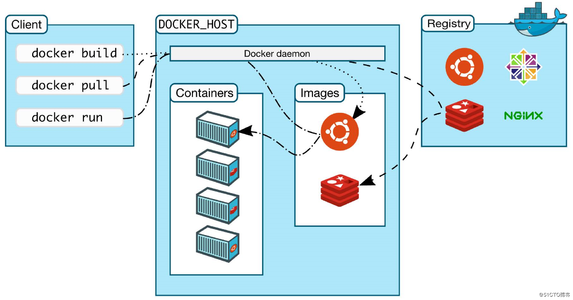
**7..更加简单的维护和拓展;** Docker使用了分层存储以及镜像技术,使得应用重复部分的复用更为容易，也使得应用的维护更新更加的简单,基于基础镜像进一步扩展镜像也非常简单;此外，docker官方提供了一系列的镜像，既可以直接在生产环境中使用,也可以作为基础进一步定制,大大降低了应用服务镜像制作成本;

更多资料：

http://www.dockerone.com/article/6051

## 2.如何使用Docker？

### 1.Docker的核心概念



a.Docker主机(host):安装了docker程序的机器(Docker直接安装在操作系统之上)

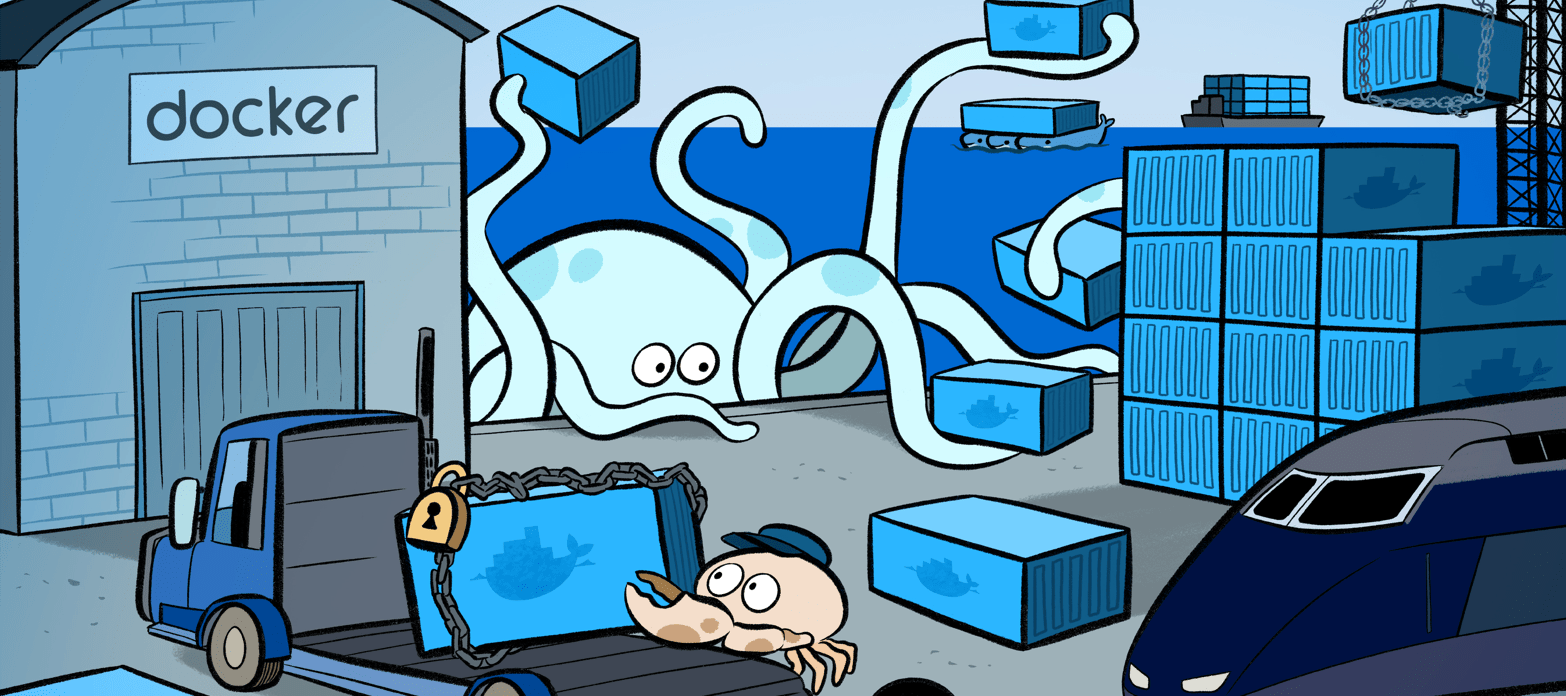
b.Docker客户端（client）：连接docker主机进行操作;

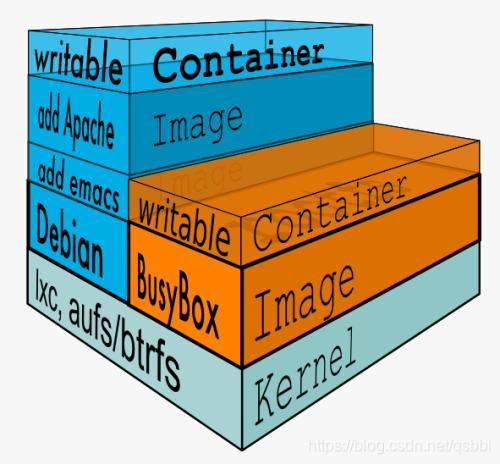
c.Docker仓库(repository)：用来保存各种打包好的软件镜像;

d.Docker镜像(Images) :软件打包好的镜像,放在docker仓库;

e.Docker容器(container)：镜像启动后的实例称为一个容器,容器是独立运行的一个或一组应用;

### **2.**Docker的设计理念





Docker的思想来源于集装箱；

在一个系统中运行多个传统应用，由于项目与项目之间的环境是交叉的，多个应用之间可能会因为端口的占用或冲突就可能报错;

Docker将每个项目打包装箱，每个箱子是互相**隔离**的,不需要担心端口冲突及环境的影响;Docker通过隔离的方式可以更好的利用服务器资源;

### 2.1Docker是如何工作的？

Docker是一个Client-Server结构系统，Docker守护进程运行在主机上，然后通过Socket连接从客户端访问，守护进程从客户端接收命令并管理运行在主机上的容器。容器，是一个运行时环境，相当于一个集装箱。





### 为什么Docker比虚拟机快

1. Docker有着比虚拟机更少的抽象层，由于docker不需要Hypervisor实现硬件资源虚拟化，运行在Docker容器上的程序之间使用宿主机的硬件资源，因此在cpu和内存的使用率上Docker在效率上具有更明显的优势。
2. Docker共享宿主机内核，而不需要Guest OS（安装在VM中的系统）;

当docker新建一个容器时，不需要想虚拟机一样加载一套完整操作系统，因此新建一个容器的速度快，只需要几秒钟；当新建一个虚拟机的时候，虚拟机软件需要加载Guest OS,加载操作系统资源的过程比较费时，一般需要数分钟;







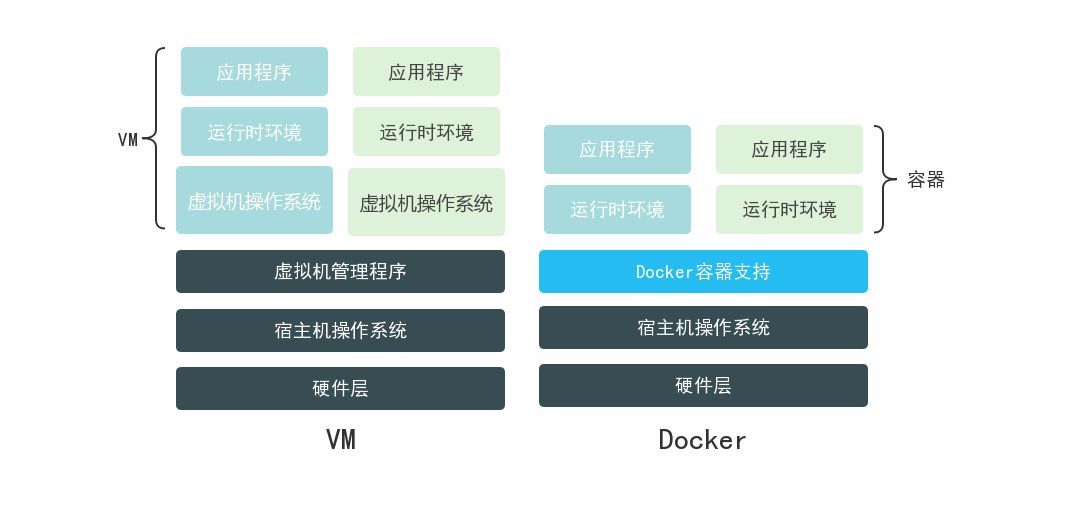












### 4.Docker使用步骤

1.安装Docker;

2.去Docker仓库找到需要安装的软件对应的镜像;

3.使用Docker运行该软件的镜像,这个镜像会产生一个Docker容器,容器就代表这个当前正在运行的应用;

4.对容器的启动停止就是对软件的启动停止;

### 5.Docker如何学习Docker

1.熟悉Linux命令及相关背景知识;

2.熟悉Maven/git方面的知识

JavaEE java springmvc/springboot/mybatis

Docker开发：go swarm/compose/Machine/mesos/k8s/CI|Cd jenkinds整合

官网地址：<https://www.docker.com/>

官方文档地址：<https://docs.docker.com/>

可以在官网底部找到入口且官方文档比较详细;

仓库地址：<https://hub.docker.com/>

B站:

<https://www.kuangstudy.com/>

<https://space.bilibili.com/95256449>

CSDN：

<https://blog.csdn.net/secowo/article/details/109595173>

网易：

<https://www.163.com/dy/article/GK23FBTE0552BFLU.html>

网易云课堂：

<https://study.163.com/course/courseMain.htm?courseId=1209483805>

简书：

腾讯课堂：

<https://ke.qq.com/course/package/35059>

慕课网：

<https://www.imooc.com/learn/867>

Runoob:

https://www.runoob.com/docker/docker-tutorial.html

百度云盘：

## 3.如何安装Docker？



### 在CentOS上安装Docker步骤

查看Linux系统的版本

cat /etc/os-release

1.查看Linux系统的内核版本

必须是3.10及以上

**uname -r**

2.升级软件包及内核;(如果Linux系统的内核版本低于3.10)

**yum update**

3.卸载旧的版本

**yum remove docker \**

**docker-client \**

**docker-client-latest \**

**docker-common \**

**docker-latest \**

**docker-latest-logrotate \**

**docker-logrotate \**

**docker-engine**

1. 设置存储库

设置 Docker 存储库之后，可以从存储库安装和更新 Docker;

a.安装yum-utils包（提供yum-config-manager 实用程序）并设置**稳定**存储库

**yum install -y yum-utils**

b.设置存储库

**yum-config-manager \**

**--add-repo \**

[**https://download.docker.com/linux/centos/docker-ce.repo**](https://download.docker.com/linux/centos/docker-ce.repo)

**由于官方存储库较慢,建议使用国内的存储库:**

3.安装 docker

**yum install docker-ce docker-ce-cli containerd.io**

安装后可以查看docker版本号

**docker -v**

4.启动docker

**systemctl start docker**

5.将docker服务设为开机启动

**systemctl enable docker**

**官方文档：**<https://docs.docker.com/engine/install/centos/>

### 在Widnows上安装Docker

官方文档：<https://docs.docker.com/desktop/windows/install/>

### 3.其他平台上安装Docker

可以参考Docker的官网，Docker官网提供了各个平台安装Docker的相关文档;

**官网教程：<https://docs.docker.com/get-docker/>**

### 4.配置 DockerHub 加速器

国内从 Docker Hub 拉取镜像速度比较慢，可以配置镜像加速器。Docker 官方和国内很多云服务商都提供了国内加速器服务:

Docker 中国官方镜像 https://registry.docker-cn.com

DaoCloud 镜像站 http://f1361db2.m.daocloud.io

Azure 中国镜像 https://dockerhub.azk8s.cn

科大镜像站 https://docker.mirrors.ustc.edu.cn

七牛云 https://reg-mirror.qiniu.com

网易云 https://hub-mirror.c.163.com

腾讯云 https://mirror.ccs.tencentyun.com

2. Linux实例配置（腾讯云为例）

1. 创建或修改 /etc/docker/daemon.json 文件，并写入以下内容：

 vim /etc/docker/daemon.json

{

"registry-mirrors": [

"https://mirror.ccs.tencentyun.com"

]

}

2. 依次执行以下命令，重新启动 Docker 服务。

$ sudo systemctl daemon-reload

$ sudo systemctl restart docker

# Ubuntu16.04 请执行 sudo systemctl restart dockerd 命令

3.检查加速器是否生效

执行 docker info 命令，返回结果中包含以下内容，则说明配置成功。

Registry Mirrors:

https://mirror.ccs.tencentyun.com

支持配置多个镜像加速器服务

{

"registry-mirrors": [

"https://mirror.ccs.tencentyun.com",

"https://registry.docker-cn.com"

]

}

## 4.如何操作Docker镜像



### 1.检索

语法：docker search [options] 关键字

说明：我们经常去docker hub上检索镜像的详细信息,如镜像的Tag;

默认去 docker hub 仓库搜索镜像; <https://hub.docker.com/>

常用项[options]:

-f，--filter filter：过滤

-s，列出收藏数不小于指定数值的镜像

--no-trunc:显示完整的镜像信息

--automated:只列出automated build 类型的镜像

实例:docker search redis

说明：搜索指定镜像信息

实例：docker search -f stars=200 mysql

    docker search --filter stars=200 mysql

说明:搜索镜像的STARS大于等于200的镜像;

实例：docker search -s 2000 --no-trunc mysql

说明:搜索镜像的STARS大于等于200的镜像并且显示完整的镜像描述;

### 2.拉取

语法：docker pull [OPTIONS] 镜像名[:TAG|@DIGEST]

说明：:tag是可选的,tag表示标签，多为软件的版本,默认为latest

实例：docker pull redis

说明：拉取指定镜像

### 3.列表

语法：docker images [options] [repository[:tag]]

可选项[options]：

-a, --all ：查看所有本地镜像包括中间镜像;

-q, --quiet :只显示镜像的ID

--digests: 显示镜像的摘要信息

--no-trunc:显示完整的镜像信息

结果说明:

REPOSITORY: 镜像的仓库源

TAG: 镜像的标签

IMAGE ID:镜像的编号

CREATED:镜像的创建时间

SIZE：镜像的大小

实例：docker images

说明：查看所有的本地镜像

实例：docker images redis

说明：查看本地镜像中REPOSITORY为redis的镜像

### 4.删除

语法：docker rmi [options] image [image...]

说明：删除指定的本地镜像，多个用空格隔开

image:镜像的名称或镜像名称(REPOSITORY)

常用项[options]:

-f:强制删除

实例 docker rmi snowdreamtech/frps(镜像名称)

docker rmi 24207ccc9cce(镜像ID)

说明：删除指定镜像

实例：docker rmi nginx tomacat

说明：删除多个镜像

实例：docker rmi -f $(docker images -aq)

说明：删除本地所有的镜像

## **5.如何操作Docker容器**





软件镜像–运行镜像—产生一个容器（正在运行的软件）

### 运行



**语法:docker run [options] image [command] [arg...]**

常用option选项:

-d:后台运行容器即创建一个守护式容器在后台一直运行

-i:表示以交互模式运行容器, 通常与-t同时使用;

-t:表示容器启动后会进入命令行,加上这个参数后，容器创建就能登录进去，即分配了一个伪终端.

-p:端口映射,将主机端口映射到容器内部的端口(指定容器端口)

语法：-p 主机端口：容器内部端口 (-p 6379:6379 )

-p ip:主机端口：容器内部端口

-p ip:容器端口

-p 容器端口

-P:随机端口

-v:给容器挂载存储卷，挂载到容器的某个目录 宿主机目录：容器目录 (将宿主机的目录挂载到容器中)(将主机目录与容器目录之间进行共享)

--name:指定容器名字,后续可以通过名字进行容器管理

--restart=always:重启docker时，自动启动相关容器

--privileged=true 容器内部对挂载的目录拥有读写等特权

### 列表

语法：docker ps [options]

说明：查看本地所有容器，默认查询所有运行中的容器

常用项：

-a 查看所有容器(包括历史运行过的容器)

-l 显示最近创建的容器

-n 显示最近创建的N个容器

-q 静态模式，只显示容器编号

--no-trunc 不截断输出

实例：docker ps

说明：默认查看运行中的容器)

实例：docker ps -a

说明：加上-a 可以查看所有容器（包括历史运行过的容器）

实例：docker ps -n 2

docker ps -n=2

说明：查看最近创建的2个容器

### 启动

语法： docker start container-name/container-id

说明：启动容器

### 停止

语法：docker stop container-name/container-id

说明：停止当前运动的容器

### 删除

命令：docker rm container-id

说明：删除指定的未启动的容器

### 2端口映射

语法：-p 主机端口：容器内部端口 (-p 6379:6379 )

docker run -d -p 6379:6379 -name myredis docker.io/redis

说明：-p:主机端口(映射到)容器内部的端口

### 容器日志

命令：docker logs [option] container

说明：查看容器日志

container：容器名称|容器ID

常用项[options]:

-f:动态查询日期

-t:加入时间戳

--tail ：显示最后多少条

常用实例：docker logs -f container-name/container-id

说明： -f动态查询日志

### 7.1查看容器内运行的进程

语法：docker top 容器id或容器名称

### 7.2查看容器细节

语法：docker inspect 容器ID或容器名称

### 进入正在运行中的容器并以命令行交互

语法：docker exec [options] container command [arg...]

说明：进入容器

container ：容器名称或容器ID

在容器中打开新的终端，并且可以启动新的进程

语法：docker attach 容器id或容器ID

说明：直接进入容器启动命令的终端，不会启动新的进程

常用实例： docker exec -it container-id baseShell

### 容器与主机之间数据拷贝;

语法：docker cp [options] 容器名称或容器ID:容器目录或文件 宿主机目录或文件

宿主目录：

### 将容器的文件更改或设置提交到新映像（用于定制自己的镜像）

语法：Docker commit [options] CONTAINER

实例：docker commit -m=”提交的描述信息” -a=”作者” 容器id或名字 创建的目标镜像[:tag]

## Docker镜像概述

### 什么是镜像？

镜像是一种轻量级、可执行的独立软件包,用来打包软件运行环境和基于运行环境开发的软件,它包含运行某个软件所需的所有内容，包括代码、运行时环境、库、环境变量和配置文件。

### 什么是联合文件系统(UnionFS)?

Union文件系统(UnionFS)是一种分层、轻量级并且高性能的文件系统，它支持文件系统的修改作为一次提交来一层层的叠加，同时可以将不同目录挂载到同一个虚拟文件系统下(unite several directories into a single virtual filesysem)。Union文件系统是Docker镜像的基础。镜像可以通过分层来进行继承，基于基础镜像(没有父镜像),可以制作各种具体的应用镜像。

特性：

一次同时加载多个文件系统,但从外面看起来，只能看到一个文件系统，联合加载会把各层文件系统叠加起来，这样最终的文件系统会包含所有底层的文件和目录；

### 镜像的特点

Docker的镜像都是只读的，当容器启动时，一个新的可写层加载到镜像的顶部。这一层通常被称作“容器层”,”容器层”之下的都叫”镜像层”

### Docker镜像加载的原理

Docker的镜像实际上由一层一层的文件系统组成，这种层次级的文件系统UnionFs。

Bootfs(boot file system)主要包含了bootloader和kernel,bootloader主要是引导加载kernel,linux刚启动时会加载bootfs文件系统。在docker镜像的最底层是bootfs。这一层和我们典型的Linux/unix系统是一样的，包含boofs加载器和内核。当boot加载完成之后整个内核就都在内存中了，此时内存的使用权已由boofs转交给内核，此时系统也会卸载bootfs。

Rootfs(root file system)，在bootfs之上。包含的就是典型Linux系统中的/dev,/proc,/bin,/etc等标准目录和文件。Rootfs就是各种不同的操作系统发行版，比如ubunt,Centos等等;

### 平时 我们安装进虚拟机的CentOS都是好几个G,为什么Docker这里才200M?

对于一个精简的OS，rootfs可以很小，只需要包括最基本的命令、工具和程序库就可以了,因为底层直接用Host的kernel,自己只需要提供rootfs就行了。有此可见对应不同的linux发行版，bootfs基本是一致的，rootfs会有差别，因此不同的发行版可以共用bootfs。

### Docker镜像分层结构有什么好处？

最大的好就是共享资源。多个镜像从相同的base镜像构建而来，那么宿主机只需要在磁盘上保存一份base镜像，同时内存中也只需要加载一份base镜像，就可以为所有的容器服务了。而且镜像的每一层都可以被共享。

## 如何访问Docker仓库

1. 

### 1.Docker hub公共镜像

Docker Hub Docker 官方提供的最大的公共镜像仓库，地址为https: //hub.docker.com，大部分对镜像的需求，都可以通过在 Docke Hub 中直接下载镜像来现;

### 2.第三方镜像市场

1. 国内不少云服务商都提供了 Docker 镜像市场（腾讯云、阿里云等）

### 3.如何搭建本地私有仓库

1. 使用 registry 创建私有仓库
2. 管理私有仓库

## Docker如何管理数据



### 容器中数据管理的主要方式有哪些？

容器中的管理数据主要有两种方式

数据卷(Data Volumes) ：容器内数据直接映射到本地主机环境；

数据卷容器（Data Volume Containers ）:使用特定容器维护数据卷;

### 什么是数据卷？

数据卷( Data Volumes) 个可供容器使用的特殊目录，它将主机操作系统目录直接 映射进容器，类似于 Linux 中的 mout 行为。

### 数据卷有什么特性？

1. 数据卷可以在容器之间共事和重用，容器间传递数据将变得高效与方便；
2. 对数据卷内数据的修改会立马生效，无论是容器内操作还是本地操作；
3. 对数据卷的更新不会影响镜像，解耦开应用和数据；
4. 卷会一直存在 ，直到没有容器使用，可以安全地卸载它;

### 如何创建卷

语法：docker volume create [options] [volume]

说明：创建卷

常用项:

-d 指定驱动名称

实例：docker volume create -d test

语法：docker volume ls [options]

说明：列出已有的数据卷

语法：docker volume inspect [options] volume [volume...]

说明：显示一个或多个卷的详细信息

语法：docker volume prune [options]

说明：删除所有未使用的本地卷

语法：docker volume rm [options] volume [volume...]

说明：删除一个或多个卷。无法删除容器正在使用的卷。

常用项：

-f, --force 强制删除一个或多个卷

### 如何绑定卷

除了使用Volume子命令来管理数据卷外，还可以在创建容器的时候将主机本地的任意路径挂载到容器内作为数据卷，这种形式创建的数据卷被称为绑定数据卷。

在使用docker run [options] image [command] [arg...]命令的时候，可以使用-mount选项来使用数据卷；

--mount 选项支持三种类型的数据卷：

Volume:普通数据卷，映射到主机 /var/lib/docker/volumes路径下；

Bind：绑定数据卷，映射到主机指定路径下;

Tmpfs:零时数据卷,只存在内存中。

实例：docker run -d -p 9093:9093 \

--name itour-online-web \

–mount type=bind,source=/webapp,destination=/opt/webapp \

ben/itour-online-manager-web

等同于

docker run -d -p 9093:9093 \

--name itour-online-web \

–v webapp:opt/webapp \

ben/itour-online-manager-web

其中本地目录必须为绝对路径，容器内的路径可以为相对路径，如果路径不存在，docker会默认创建。

### 什么是数据卷容器？

如果需要多个容器之间共享一些持续性更新的数据，最简单的方式是使用数据卷容器。数据卷容器也是一个容器，但是它的目录是专门提供数据卷给其他容器挂载。

--volumes-from List (数据卷容器)

### 数据卷容器有什么特性？

### 如何创建数据卷容器

### 如何绑定数据卷容器

### 如何利用数据卷容器迁移数据？

#### 备份

#### 恢复

### 端口映射与容器互联？

在启动容器的时候， 如果不指定对应参数， 在容器外部是无法通过网络来访问容器内的网络应用和服务的。

当容器中运行一些网络应用， 要让外部访问这些应用时， 可以通过-P或-p参数来指定端口映射。

当使用P（大写的）标记时， Docker 会随机映射一个 49000~49900 的端口到内部容器开放的网络端口；

-p (小写的）则可以指定要映射的端口，并且，在一个指定端口上只可以绑定 一个容器。 支持的格式有

映射到指定地址的指定端口

**IP:HostPort:ContainerPort (**-p 127.0.0.1:5000:5000**)**

映射到指定地址的任意端口

**IP:: ContainerPort (**-p 127.0.0.1::5000**)**

映射所有接口地址:

**HostPort: ContainerPort (-p 9000:9000)**

**多次使用-p标记可以绑定多个端口,**

### 端口映射实现容器访问?

容器的互联(Iinking)是一种让多个容器中的应用进行快速交互的方式。它会在源和接收容 器之间创建连接关系，接收容器可以通过容器名快速访问到源容器，而不用指定具体的IP地址;

#### 1自定义容器命名

使用--name标记可以为容器自定义命名

使用docker ps来验证设定的命名

也可以使用docker inspect [options] name|id [name|id...]查看

容器的名称是唯一的。如果巳经命名了一个叫web的容器，当你要再次使用web这 个名称的时候，需要先用docker rm命令删除之前创建的同名容器。

在执行docker [con 七ainer] run的时候如果添加--rm标记，则容器在终止后会 立刻删除。注意，- -rm 和-d参数不能同时使用。

#### 容器互联

使用--link参数可以让容器之间安全地进行交互。

--link参数的格式为--link name: alias, 其中name是要链接的容器的名称 ， alias是别名。

使用 docker ps 来查看容器的连接：

### 互联机制实现便捷互访？

## 如何使用Dockerfile构建镜像；



### 什么是Dockerfile？

Dockerfile 是一个文本格式的配置文件， 使用 Dockerfile 可以快速创建自定义的镜像。

### Dockerfile文件的基本结构

Dockerfile 由一行行命令语句组成， 并且支持以＃开头的注释行。 一般而言， Dockerfile 主体内容分为四部分：基础镜像信息、 维护者信息、 镜像操作指 令和容器启动时执行指令。

### Dockerfile文件中的指令

Dockerfile 中指令的一般格式： 指令 参数；

配置指令：

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 格式 | 作用 | 说明 |
| FROM | FROM IMAGE [:TAG|@DIGEST] | 指定所创建镜像的基础镜像 | 任何 Dockerfile 中第一条指令必须为FROM 指令。 并且， 如果在同一个Dockerfile 中创 建多个镜像时， 可以使用多个 FROM 指令 |
| VOLUME | VOLUME ["/data”] | 创建一个数据卷挂载点 | 运行容器时可以从本地主机或其他容器挂载数据卷， 一般用来存放数据库和需要保持的 数据等 |
| ENTRYPOINT | ENTRYPOINT ["executable", "paraml ", "param2"]:  exec 调用执行；  ENTRYPOINT command param 1 param2: shell 中执行。  此时， CMD指令指定值将作为根命令的参数. | 指定镜像的默认入口命令,该入口命令会在启动容器时作为根命令执行， 所有传人值作 为该命令的参数。 | 每个 Dockerfile 中只能有一个 ENTRYPOINT, 当指定多个时， 只有最后一个起效。 在运行时， 可以被 --entrypoint 参数覆盖掉; |

操作指令：

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 格式 | 作用 | 说明 |
| RUN | 1.RUN <COMMAND>  2.RUN [“excutabel”,”param1”,”param2”] | 运行指定命令 | 第一种指令格式默认将在shell终端中运行命令，即/bin/sh -c;第二种方式则使用exec执行，不会启动shell环境。  第二种指定格式会被解析为JSON数组，因此必须用双引号。  使用其他终端可以使用第二种指令方式如  RUN [”/bin/bash" , ” -C” ,”echo h e llo”]  每条run指令将在当前镜像的基础上执行指定命令，并提交为新的镜像层。当命令较长是可以使用\来换行 |
| ADD |  | 添加内容到镜像 |  |
| COPY |  | 复制内容到镜像 |  |
| CMD |  | 启动容器时指定默认执行的命令 |  |

### 创建镜像

1. **使用Docker build 构建镜像**

编写Dockerfile文件后可以使用docker build命令创建镜像;

语法：docker build [OPTIONS] PATH | URL | -

说明：该命令将读取指定路径下(包含子目录)，的Dockerfile，并将该路径下所有数据作为上下文(Context)发送给Docker服务端。Docker服务端在校验Dockerfile格式通过后，逐条执行其中定义的指令,碰到Add、copy和run命令会生成一层新的镜像。最终如果创建镜像成功，会返回最终镜像的ID；

如果上下文过大，会导致发送大楼的数据给服务器，延缓创建过程。因此除非是生成镜像所需必需的文件，不然不要放到上下文路径下。如果使用非上下文路径下的dockerfile，可以通过-f选项来指定其路径。

要指定生成镜像的标签信息，可以通过-t选项。该选项可以重复使用多次为镜像一次添加多个名称。

1. **选择父镜像**

大部分情况下，生成新镜像都需要通过FROM指令来指定父镜像。父镜像是生成镜像的基础。会直接影响到所生成镜像的大小和功能。

可以选择两种镜像作为父镜像，一种是基础镜像(baseImage),另外一种是普通镜像(一般由第三方基于基础镜像创建)

基础镜像比较特殊，dockerfile中往往不存在Form指令，或者基于scratch镜像(FROM scratch);

**使用.dockerignor文件**

**可以通过.dockerignore文件(每一行添加一条匹配模式),来让docker忽略匹配路径或文件，才创建镜像时候不将无关数据发送到服务端。**

**多步骤创建：**

**Docker支持多步骤创建镜像，可以精简最终生成的镜像大小;**

### 总结

1. 精简镜像用途；尽量让每个镜像的用途比较集中单一，避免构造大而复杂、多功能的镜像;
2. 选用合适的基础镜像;
3. 提供注释和维护者信息
4. 正确的使用版本号：使用明确的版本信息，而非依赖于默认的latest。
5. 减少镜像的层数;尽量合并run、add、copy指令；
6. 恰当使用多步骤创建：
7. 使用.dockerignore文件：.dockerignore文件可以标记在执行docker build时忽略的路径和文件，避免发送不必要的数据内容，从而加快整个镜像创建过程。
8. 及时删除临时文件和缓存文件；
9. 提高生成速度;
10. 调整合理的指令顺序:
11. 减少外部源的干扰;

# 实际应用

## 如何使用Docker安装和使用常见的操作系统

### BusyBox

BusyBox 是一个集成了一百多个最常用 Linux 令 的精简工具箱，它只有不 2MB 大小。

BusyBox 官网：<https://busybox.net>

官方仓库 <https://git.busybox.net/busybox/>

官方镜像 <https://hub.docker.com/_/busybox>

官方镜像仓库 <https://github.com/docker-library/busybox/>

### Alpine

Alpine 操作系统是一个面向安全的轻型 Linux 发行版。它不同于通常 Linux 发行版，Alpine 采用了 musl libc 和 busybox 以减小系统的体积和运行时资源消耗，但功能上比 busybox 又完善的多，因此得到开源社区越来越多的青睐。在保持瘦身的同时，Alpine 还提供了自己的包管理工具 apk，可以通过 https://pkgs.alpinelinux.org/packages 网站上查询包信息，也可以直接通过 apk 命令直接查询和安装各种软件。

相关资料：

Alpine官网：<https://alpinelinux.org/>

官方仓库：<https://github.com/alpinelinux>

官方镜像: <https://hub.docker.com/_/alpine>

官方镜像仓库：<https://github.com/gliderlabs/docker-alpine>

使用帮助：<https://yeasy.gitbook.io/docker_practice/os/alpine>

### Debian/Ubuntu

Debian和Ubuntu都是目前较为流行的debian系的服务器操作系统。Docker hub提供了官方镜像，国内各大容器云服务都提供了完整的支持。

Debian是基于GPL授权的开源操作系统，是目前个人电脑与服务器中最受欢迎的开源操作系统之一。

Debian官网：<https://www.debian.org/>

Debian官方镜像：<https://hub.docker.com/_/debian>

Ubuntu是以桌面应用为主的GnUL/Linux开源操作系统，

Ubuntu 官网：<https://ubuntu.org.cn/global>

Ubuntu 官方镜像：<https://hub.docker.com/_/ubuntu>

### CentOS/Fedora

CentO和Fedora都是基于Redhat的Linux发行版。CentOS是目前企业级服务器的常用操作系统，Fedora则主要面向个人桌面用户。

Fedora相关资料：

Fedora官网：<https://getfedora.org/>

Fedora官方镜像：<https://hub.docker.com/_/fedora>

CentOS资料：

CentOS官网：<https://centos.org/>

CentOS官方镜像：<https://hub.docker.com/_/centos>

## 2.如何为镜像添加SSH服务

通常，系统管理员习惯通过SSH服务来远程登录管理服务器，但是Docker的很多镜像是不带SSH服务的，如何才能管理容器呢？如何解决远程管理容器的问题呢？当我们需要远程登录到容器内部进行一系列操作的时候，就需要SSH支持了。

### 如何创建一个带有SSH服务的镜像

#### 使用Docker commit 命令创建

1. 获取镜像，创建一个容器
2. 检查软件源，并使用apt-get update 命令来更新软件源信息；
3. 选择主流的openssh-server作物服务端。
   1. apt-get install openssh-server
   2. 如果需要正常启动SSH服务，则目录/var/run/sshd必须存在。

mkdir -p /var/run/sshd

3.3启动ssh服务

/usr/sbin/sshd -D

3.4查看容器的22端口(SSH服务器默认监听的端口)是否处于监听状态；

3.5 修改SSH服务的安全登录配置，取消pam登录限制

3.6在root用户下创建.ssh目录，并复制需要登录的公钥信息(一般为本地主机用户目录下的.ssh/id\_rsa.pub文件，可有ssh-keygen -t rsa 命令生成)到authorized\_key文件中

3.7创建自动启动SSH服务的可执行文件run.sh，并添加可执行权限；

3.8最后退出容器

4.保存镜像，将所退出的容器使用docker commit 命令保存一个新的镜像；

5.使用镜像，添加端口映射到22端口（容器的SSH服务监听端口）

6.可以在宿主主机或其他云主机上，通过SSH访问指定端口来登录容器;

#### 使用Dockerfile创建

1. 创建工作目录

Mkdir sshd\_ubuntu

在其中创建Dockerfile文件和run.sh文件

Touch dockerfiles run.sh

1. 编写run.sh脚本和authorized\_keys文件
2. 编写Dockerfile文件
3. 使用Docker build 命令来创建镜像
4. 测试镜像，运行容器

## 3.如何使用Docker运行常见的web服务器及应用

### Apache

### Nginx

### Tomcat

### Jetty

### Lamp

### 持续开发和管理

#### Jenkins

##### 什么是Jenkins

Jenkins是一个广泛应用的持续集成和持续交付的工具。作为开源软件项目，它旨在提供一个开放易用的持续集成平台。

##### Jenkins能做什么？

Jenkins能实现实时监控集成中存在的错误，提供详细的日志日志文件和提醒功能，并用图表的形式形象地展示项目构建的趋势和稳定性

##### Jenkins的特点

安装配置简单、支持详细的测试报表、分布式构建等；

Jenkins2.0，推出了“pipeline as code”,帮助实现对CI和CD更好的支持。通过pipline，将原本独立运行的多个任务连接起来，可以实现复制的发布流程

### GitLab

##### 什么是gitlab

GitLab是一款开源代码管理系统，支持基于git的源码管理、代码评审、issue跟踪、活动管理、wiki页面、持续集成和测试等功能。

相关资料：

Jenkins官网：<https://www.jenkins.io/>

Jenkins官方仓库 <https://github.com/jenkinsci/jenkins>

Jenkins官方镜像：<https://hub.docker.com/_/jenkins>

Jenkins官方镜像仓库<https://github.com/jenkinsci/docker>

GitLab官网：<https://github.com/gitlabhq/gitlabhq>

GitLab 官方镜像：<https://hub.docker.com/r/gitlab/gitlab-ce>

## 如何使用Docker运行常见的数据库应用

### Mysql;

docker run -d -p 3306:3306 \

--name ben-backup-mysql \

--restart=always \

-e TZ=Asia/Shanghai \

-e MYSQL\_ROOT\_PASSWORD=mysql@958958 \

-v /opt/data/mysql\_backup/mysqldata:/var/lib/mysql \

-v /etc/mysql/mysql\_backup/conf.d:/etc/mysql/conf.d \

mysql \

--lower-case-table-names=1

### Oracle

### MongoDB

### Redis

docker run -d -p 6397:6379 \

--restart=always \

--name ben-backup-redis \

-v /opt/data/redis\_backup:/data \

-v /etc/redis/redids-backup/conf/redis.conf:/usr/local/etc/redis/redis.conf \

redis \

--requirepass 'redis@958958'

### Cassandra

## 5.如何使用Docker部署和使用分布式处理与大数据平台下的常见工具

## 5.1如何使用Docker安装常见的Docker管理工具

### 1. portainer-ce

docker run -d -p 9000:9000 \

--name portainer \

--restart=always \

-v /var/run/docker.sock:/var/run/docker.sock \

-v /opt/data/portainer\_data:/data \

portainer/portainer-ce

## 如何使用Docker部署和使用分布式处理和大数据平台相关工具

#### Hadoop

#### Spark

#### Storm

#### Elasticsearch

## 如何使用Docker部署主流编程语言的开发

Java

Python

JavaScript

Go

C/C++

## 7.容器与云服务

## 8.作为开发人员我们应该如何看待容器？

## 9使用Docker过程中遇到问题及解决方案

## 如何使用Docker部署Springboot项目

1.在Linux上只安装Docker;

2切换到宿主机 :/usr/local/docker 如果没有docker就创建一个新的目录mkdir docker;

3.创建一个存放项目的文件夹 mkdir itour;

4.进入文件夹 cd /usr/local/docker/itour/

5.使用rz命令 上传springboot项目的jar;

如果找不到该命令，安装rz命令：yum install lrzsz

6.新建Dockerfile文件

Dockerfile文件,类似于window中bat文件，可以执行一些命令;

#Docker image for springboot file run

#基础镜像使用java

FROM java:8

#作者

MAINTAINER wangtao <wwangtaoc11@163.com>

#指定临时文件目录为/tmp

VOLUME /tmp

#将jar包添加到容器中并更名为itour.jar

ADD itour-online-web.jar itour.app

#运行jar

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/itour.jar"]

6.构建容器：

docker build -t itour .

7.启动容器

docker run -name itour -p 8899:8899 itour

8.运行成功及可以访问

相关文档:

<https://www.jianshu.com/p/397929dbc27d>

<https://www.imooc.com/article/44582>

Dockerfile相关文档：

<https://www.runoob.com/docker/docker-dockerfile.html>

<https://blog.csdn.net/cxzhq2002/article/details/91371225>

<https://www.cnblogs.com/edisonchou/p/dockerfile_inside_introduction.html>

<https://www.cnblogs.com/panwenbin-logs/p/8007348.html>

## 如何使用Docker部署war项目

1.Linux安装docker

1. 运行tomcat

docker run -d --name tomcat-8080 -p 8080:8080 -v /usr/local/webapps/:/usr/local/tomcat/webapps/ tomcat

2.切入到/usr/local/webapps/

3.使用rz命令选择war包;

4.浏览器访问

## Eclisp或Idea如何集成Docker实现一键部署？

### 1.Linux环境准备Cetos7(腾讯云)

a.安装Docker

b.安装Mysql

c.安装Redis

### 2.windows 环境准备

a.安装开发工具

b.安装docker插件

c.Linux远程连接工具(xshell)

d.项目准备(Springboot+Maven+mysql+Redis+eureka)

e.打包项目推送镜像到服务器

f.运行镜像

d.浏览器访问

### 3.Docker插件的安装

Idea安装Docker插件

a.打开idea,从File-->Settings-->Plugins-->Install jetBrains plugin 进入插件安装界面;

b.在搜索框中输入docker，可以看到Docker integration,点击右边的install按钮进行安装，安装后重启idea;

**Idea配置docker**

a.配置docker，连接到远程的docker容器

从File-->Setttings-->Builde,Excution,Deployment->Docker打开配置界面;

eclipse 安装Docker插件

a.Help-->Eclipse Marketplace-->搜索 Eclipse Docker Tooling

**eclipse配置docker**

1. window-->show view-->other-->选择docker explorer

相关文档：<https://www.e-learn.cn/topic/1883011>

### 4.Docker插件连接到Docker

1.Docker引擎默认不允许外部访问需要开启远程访问;  
#修改Docker服务文件  
vim /usr/lib/systemd/system/docker.service  
#修改ExecStart这行,在后面追加红色部分

ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock  
将文件内的ExecStart注释，新增上行红色字体部分;

#重新加载配置文件  
systemctl daemon-reload  
#重启服务  
systemctl restart docker.service  
#查看端口是否开启  
netstat -nlpt #如果找不到netstat命令，可以进行安装 yum install net-tools  
#检查curl是否生效，注意，必须关闭防火墙或腾讯云安全组开放2375端口，这个端口也可以自定义  
curl http://服务器地址：2375/info  
注意ip地址必须和服务器地址统一;  
相关文档：<https://blog.csdn.net/boling_cavalry/article/details/100049996>

### 5.配置Docker插件

两种方式:

1.直接在pom.xml中配置;

2.在pom中指定Dockfile文件的位置;

直接在pom.xml中配置的方式:

<!-- docker 插件 可以帮助我们在Maven工程中通过简单的配置，自动生成镜像并推送到仓库中-->

<plugin>

<groupId>com.spotify</groupId>

<artifactId>docker-maven-plugin</artifactId>

<version>1.0.0</version>

<configuration>

<!-- 镜像名称 -->

<imageName>${docker.image.prefix}/${project.artifactId}</imageName>

<!-- 指定标签 -->

<imageTags>

<imageTag>latest</imageTag>

</imageTags>

<!-- 基础镜像java 1.8 -->

<baseImage>java</baseImage>

<!-- 制作者提供本人信息 -->

<maintainer>wangtao wwangtaoc11@163.com</maintainer>

<!-- 切换到/root目录 -->

<workdir>/ROOT</workdir>

<cmd>["java","-version"]</cmd>

<entryPoint>["java","-jar","${project.build.finalName}.jar"]</entryPoint>

<!-- 指定远程docker api 地址 -->

<dockerHost>http://106.54.162.159:2375</dockerHost>

<!-- 指定Dockerfile文件的位置

<dockerDirectory>src/main/docker</dockerDirectory> -->

<resources>

<resource>

<targetPath>/ROOT</targetPath>

<!-- 指定需要复制的根目录:${project.build.directory}表示target目录 -->

<directory>${project.build.directory}</directory>

<!-- 指定需要复制的文件 ${project.build.finalName}.jar指的是打包后的文件-->

<include>${project.build.finalName}.jar</include>

</resource>

</resources>

</configuration>

<!-- 以上还需要通过mvn clean package docker:build 手动构建镜像，在<plugin>添加以下插件即可以

在项目打包Package的时候即可自动构建镜像,只需要run就可以访问了;

当执行mvn package时执行mvn clean package docker:build

-->

<executions>

<execution>

<id>build-image</id>

<phase>package</phase>

<goals>

<goal>build</goal>

</goals>

</execution>

</executions>

</plugin>

### 6.执行 mvn package [-P test]

## 13.Docker部署项目中遇到的问题及解决方案



# 开源项目

## Docker Machine

### 1.什么是Docker Machine

Machine 项目是 Docker 官方的开源项目 ，负责实现对 Docker 运行环境进行安 装和管理，特别在管理多个 Docker 环境时，使用 Mach in 要比手动管理高效得多 。

Machine 的定位是“在本地或者云环境中创建 Docker 主机” 代码在https: // github.com/docker machine 开源。

Machine 项目主要由 语言编写，用户可以在本地任意指定由 Machine 管理的 Docker 主机，并对其进行操作 。

### 2.Docker Machine能做什么？

1. 在指定节点或平台上安装 Docker 引擎，配置其为可使用的 Docke 境；
2. 集中管理（包括启动 、查看等）所安装 Dock 环境

### 3.如何安装Docker Machine？

Docker Machine 可以在多种操作系统平台上安装，包括 inux Mac OS 以及 Windows。

#### 在Linux平台安装Docker Machine

## Docker Compose

## Docker swarm